

Netcore应用

dotnet \*.dll --urls "http://\*:5001;"

dotnet \*.dll -ip 127.0.0.1 -port 5026s

dotnet WebApiCore.dll --urls=http://\*:5026 --ip=127.0.0.1 --port=5026

dotnet WebApiCore.dll --urls=http://\*:5027 --ip=127.0.0.1 --port=5027

dotnet WebApiCore.dll --urls=http://\*:5028 --ip=127.0.0.1 --port=5028

dotnet WebApiCore.dll --urls=http://\*:5029 --ip=127.0.0.1 --port=5029

dotnet WebApiProduct.dll --urls=http://\*:5126 --ip=127.0.0.1 --port=5126

dotnet WebApiProduct.dll --urls=http://\*:5127 --ip=127.0.0.1 --port=5127

dotnet WebApiCoreGateway.dll --urls=http://\*:6299 --ip=127.0.0.1 --port=6299

dotnet WebApiAuthentication.dll --urls=http://\*:7200 --ip=127.0.0.1 --port=7200

dotnet IdentityServerCore3.dll --urls=http://\*:6200 --ip=127.0.0.1 --port=6200

IdentityServerCore3.dll

Consul.exe 启动指令: Consul.exe agent -dev

Nginx启动指令：

start nginx.exe

./nginx -s stop  
./nginx -s quit  
./nginx -s reload

2.重新加载配置文件： 当 ngin x的配置文件 nginx.conf 修改后，要想让配置生效需要重启 nginx，使用-s reload不用先停止 ngin x再启动 nginx 即可将配置信息在 nginx 中生效，如下： ./nginx -s reload

dotnet WebCoreTest3.1.dll --urls=http://\*:6001 --ip=127.0.0.1 --port=6001

dotnet IdentityServerCore3.1.dll --urls=http://\*:5005--ip=127.0.0.1 --port=5005

https://localhost:5001/.well-known/openid-configuration

dotnet MvcClient.dll --urls=http://\*:5006

Redis基本指令

八种类型

string / hash / list /set /zset

List 分页第一页 热数据 数据库中经常查询的数据 很少修改的数据

Set 去重数据

Zset 带排序的权重数据

服务端

redis-server.exe redis.windows.conf

客户端

redis-cli -h 127.0.0.2 -p 6379 -a 123456

Redis 集群搭建最少6个节点

主从复制+分片

主从复制：当一个集群中的主节点挂了，则有从节点代替

分片：解决所有的读取不会所有的数据读取一台集群或者一个主节点

启动集群指令

Linux ./redis-trib.rb create --replicas 1 127.0.0.2:6379 127.0.0.2:6380 127.0.0.2:6381 127.0.0.2:6382 127.0.0.2:6383 127.0.0.2:6384

Windows redis-trib.rb create --replicas 1 127.0.0.2:6379 127.0.0.2:6380 127.0.0.2:6381 127.0.0.2:6382 127.0.0.2:6383 127.0.0.2:6384

注意127.0.0.2 ip时失败

redis-trib.rb create --replicas 1 127.0.0.1:6379 127.0.0.1:6380 127.0.0.1:6381 127.0.0.1:6382 127.0.0.1:6383 127.0.0.1:6384

客户端

redis-cli -h 127.0.0.1 -c -p 6379 -a 123456

Redis-cli -h 127.0.0.1 -c -p 6380 -a 123456

Redis-cli -h 127.0.0.1 -c -p 6381 -a 123456

Redis-cli -h 127.0.0.1 -c -p 6382 -a 123456

Redis-cli -h 127.0.0.1 -c -p 6383 -a 123456

Redis-cli -h 127.0.0.1 -c -p 6384 -a 123456

Flushall

Cluster reset

redis-trib.rb check 127.0.0.1:6379

官方是推荐使用redis-trib.rb fix 来修复集群。通过cluster nodes看到7001这个节点被干掉了。可以按照下面操作进行修复

[root@slave2 redis]# redis-trib.rb fix 127.0.0.2:6380

修复完成后再用check命令检查下是否正确（查看别的节点）

[root@slave2 redis]# redis-trib.rb check 127.0.0.2:6380

只要输入任意集群中节点即可，会自动检查所有相关节点。

可以查看相应的输出看下是否是每个Master都有了slots。

如果分布不均匀那可以使用下面的方式重新分配slot:

[root@slave2 redis]# redis-trib.rb reshard 127.0.0.2:6379

集群搭建遇到问题

port 7000 //端口7000,7002,7003

bind 本机ip //默认ip为127.0.0.1 需要改为其他节点机器可访问的ip 否则创建集群时无法访问对应的端口，无法创建集群

daemonize yes //redis后台运行

pidfile /var/run/redis\_7000.pid //pidfile文件对应7000,7001,7002

cluster-enabled yes //开启集群 把注释#去掉

cluster-config-file nodes\_7000.conf //集群的配置 配置文件首次启动自动生成 7000,7001,7002

cluster-node-timeout 15000 //请求超时 默认15秒，可自行设置

appendonly yes //aof日志开启 有需要就开启，它会每次写操作都记录一条日志

syslog-ident = redis  
Invalid argument during startup: unknown conf file parameter : cluster-config-file

我一开始以为是redis的windows服务版本不支持这个配置选项,网上各种找和查,最后才知道原来redis的配置项前面不能有空格

关于启动redis集群时：

**[ERR] Node 172.168.63.202:7001 is not empty. Either the nodealready knows other nodes (check with CLUSTER NODES) or contains some**

的问题

解决办法：

1)将每个节点下aof、rdb、nodes.conf本地备份文件删除；   
2)172.168.63.201:7001> flushdb #清空当前数据库(可省略)

dotnet ConsoleAppServer.dll --ip=127.0.0.1 --port=998

秒杀测试指令

dotnet ConsoleAppNetCore3.dll 1 56

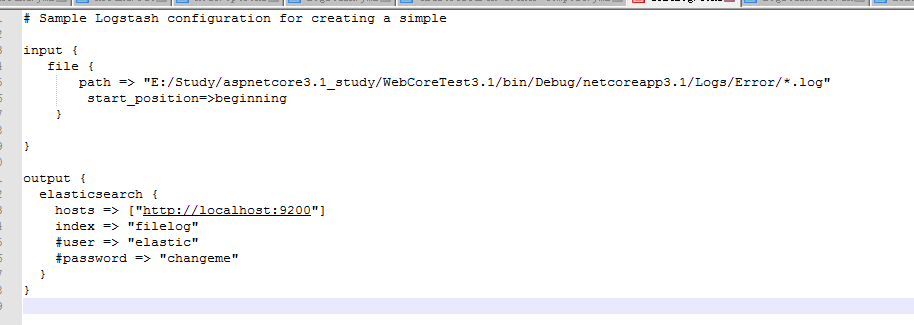
# ELK Elasticsearch Logstash Kibana

一、Logstash 启动配置：

1、指令：logstash.bat -f filelog.conf

Filelog.conf配置内容

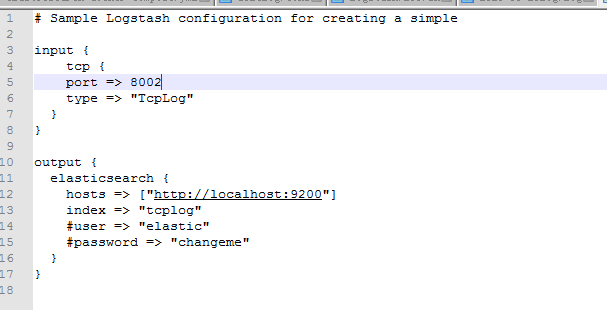
注意点：path路径斜线是 “/” 不是 “\”



2、Tcplog.conf配置内容

dotnet WebCoreTest\_ELK\_Logstash\_TCP.dll --urls=http://\*:8003 --ip=127.0.0.1 --port=8003

logstash.bat -f tcplog.conf

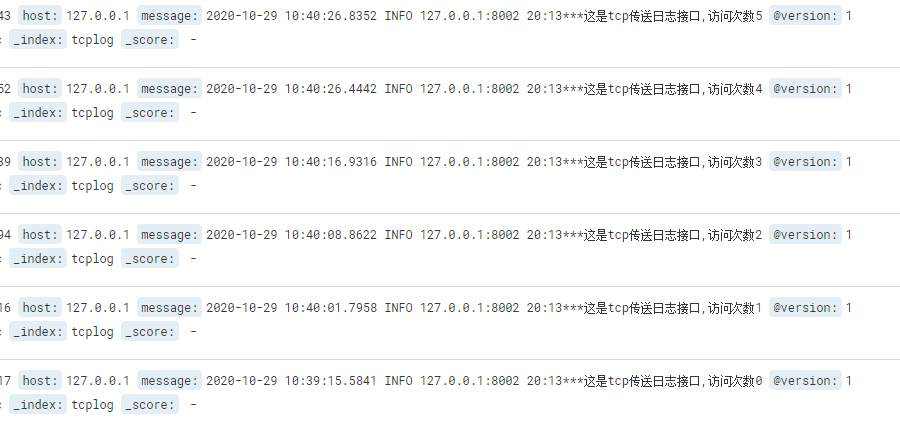


注意点：core项目配置nlog.config时配置tcp协议

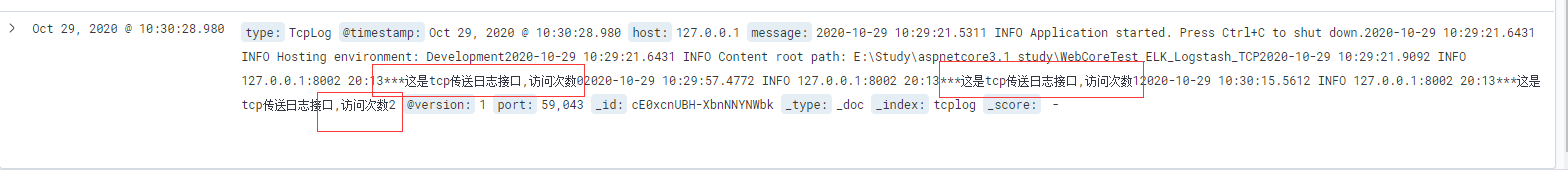
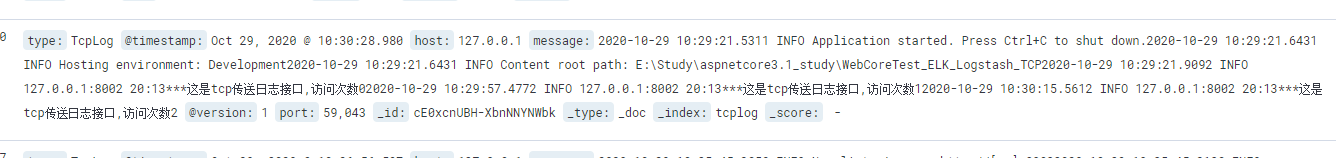
<target xsi:type="Network" name="logInfo" address="tcp://127.0.0.1:8002" layout="${longdate} ${uppercase:${level}} ${message}" keepConnection="false"/>



当属性keepConnection="false"，可以持续写入，



当设置true时，只能关闭应用后一次性写入，我第一次采坑点就是没有配置<target xsi:type="Network" name="logInfo" address="tcp://127.0.0.1:8002" layout="${longdate} ${uppercase:${level}} ${message}" />导致只能关闭应用时候一次性写入。我个人猜测是tcp端口持续暂用导致，估计八九不离十（我猜测是在本地运行core项目中应该是只能写进去一次，8002被占用，logstash应该调用不到，导致日志暂时写不进去，当core项目停止后，logstash能通过tcp传送成功）

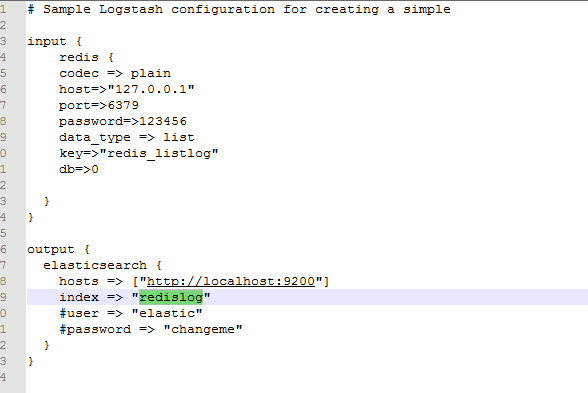


3、Redis 方式写入

logstash.bat -f redislog.conf

配置好之后按照顺序重启：redis-- es ---logstash--kibana 客户端写入正常拉取

（大数据高并发情况下日志可以采取存储到redis list中，然后配置logstash去拉取数据，logstash拉取数据后会删除，所以不用担心内存不足问题

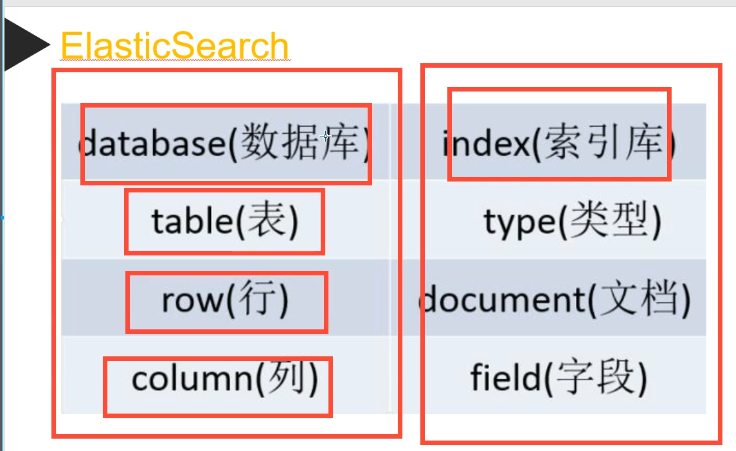
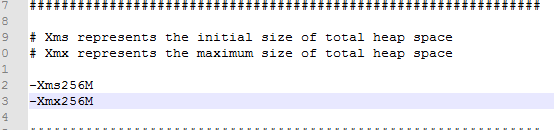


### ElasticSearch

Elasticsearch配置：

Config/jvm.options:

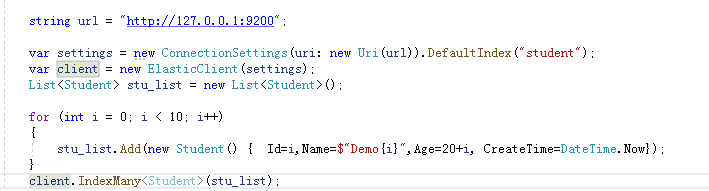
-Xms/Xmx默认是1g,改成256M



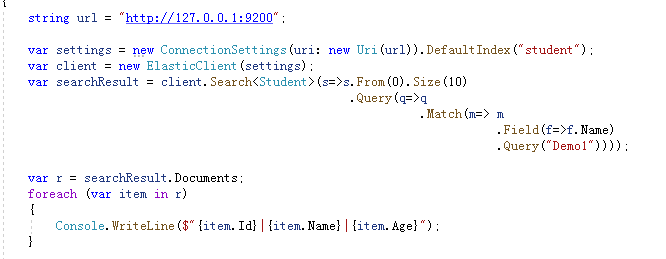
.NetCore调用需要引用NEST包

官方文档：https://www.elastic.co/guide/en/elasticsearch/client/net-api/current/index.html

插入es数据库代码



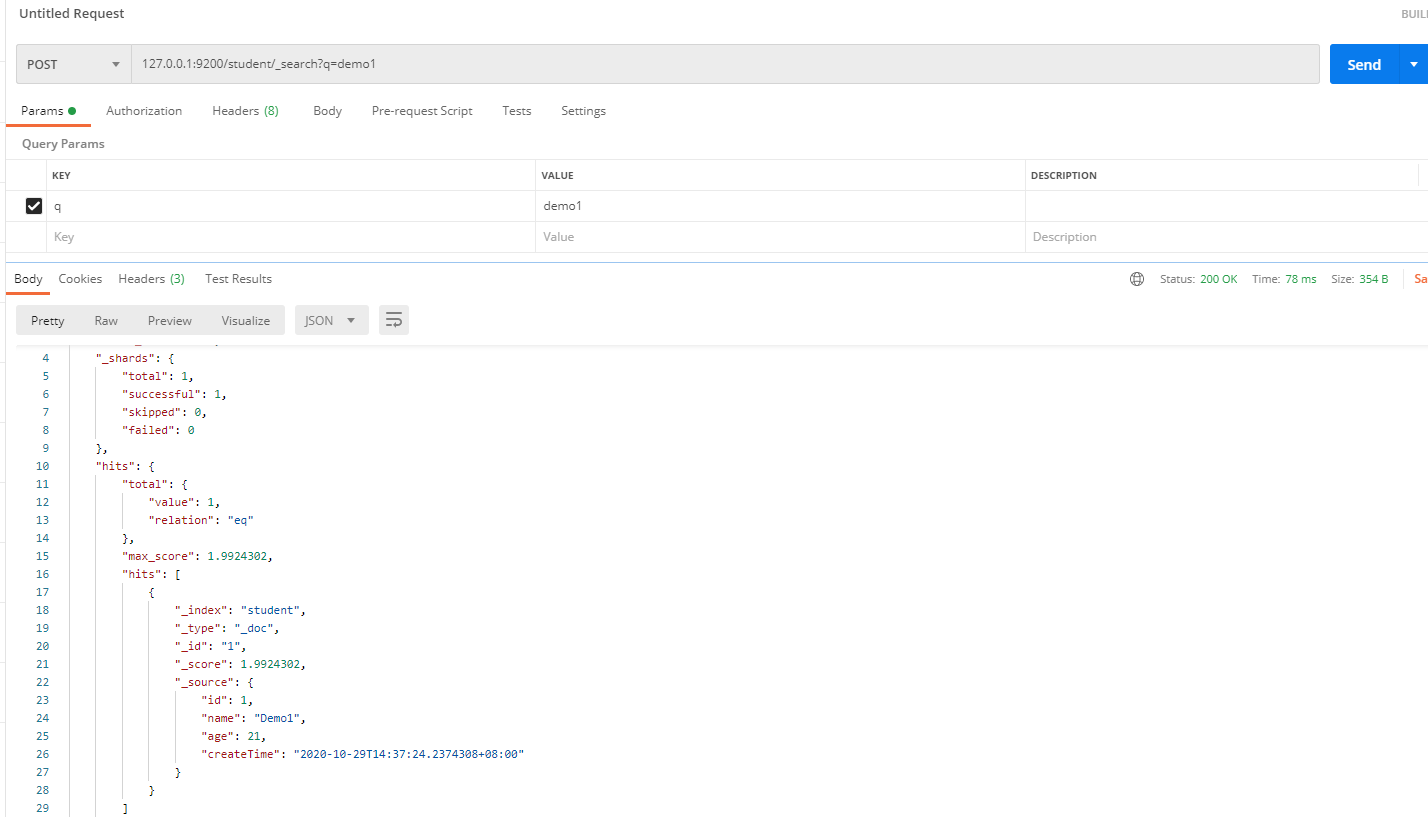
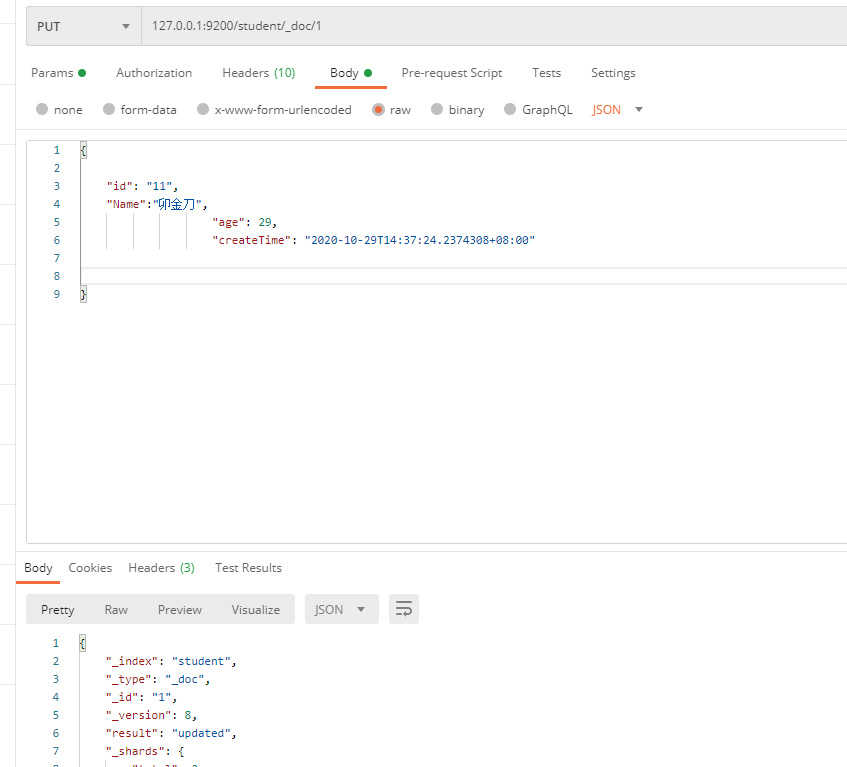
获取代码：



Postman操作

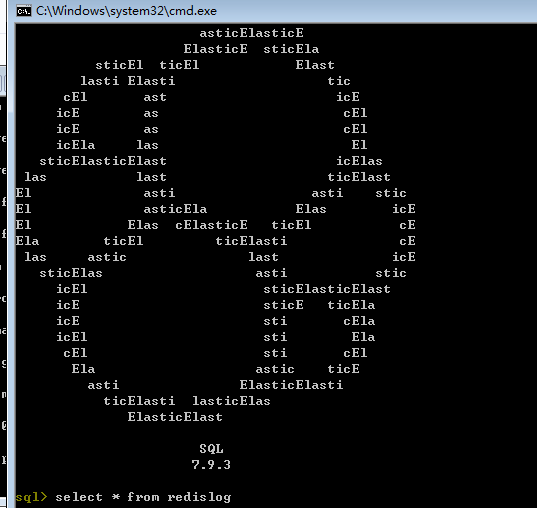
此篇博客可以参考https://blog.csdn.net/u013613428/article/details/56484794

官网文档https://www.elastic.co/guide/en/elasticsearch/reference/current/rest-apis.html



使用sql操作

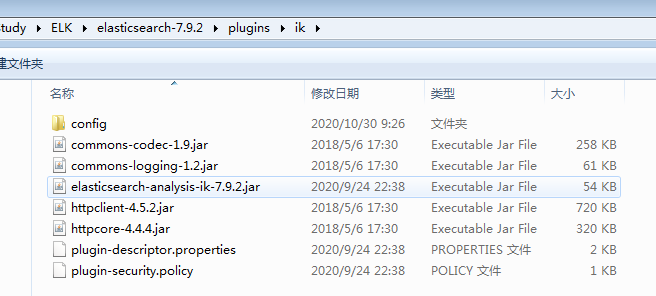
Bin/elasticsearch-sql-cli.bat



IK分词器（插件中文分词）

\elasticsearch-analysis-ik-7.9.2\bin\文件加下的分词器文件copy放到

elasticsearch下面的一个\plugins文件夹下建立一个新的文件夹\ik\文件夹下



案例：

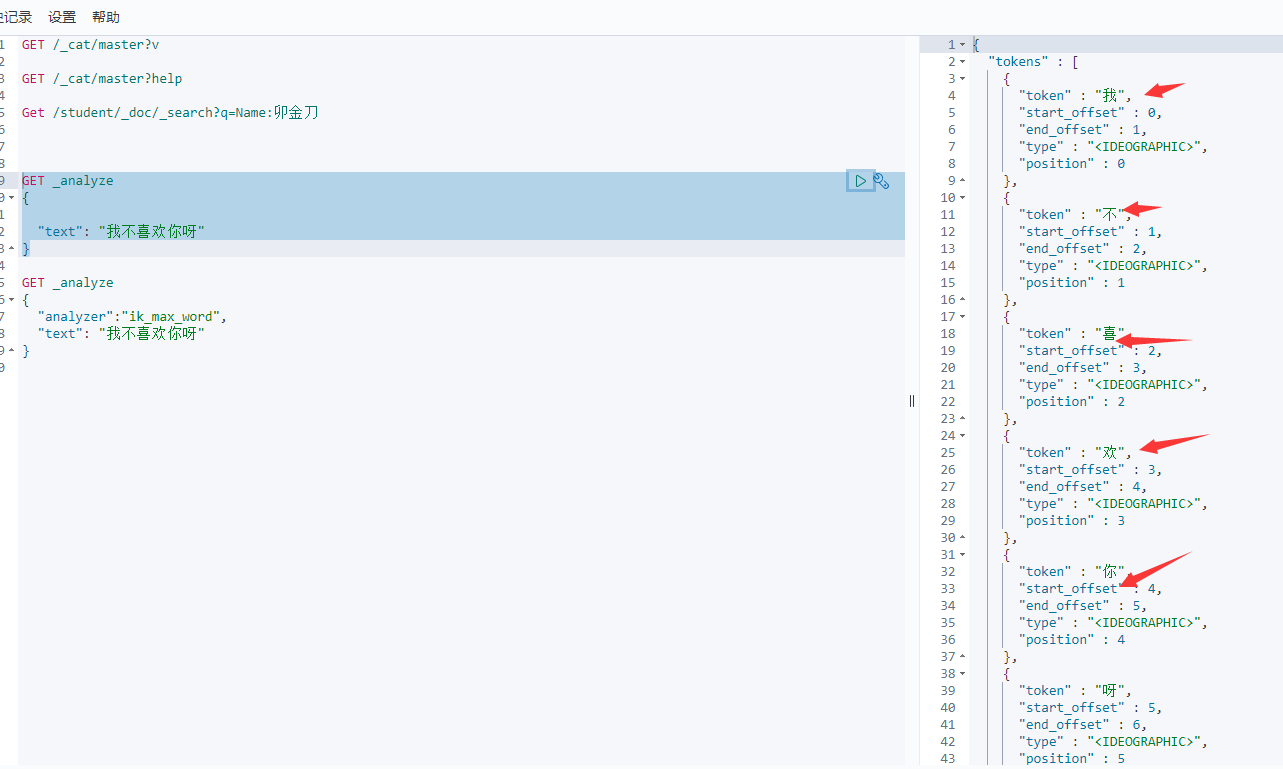
未使用分词器：

GET \_analyze

{

"text": "我不喜欢你呀"

}



使用分词器：

GET \_analyze

{

"analyzer":"ik\_max\_word",

"text": "我不喜欢你呀"

}



1. Kibana

使用前先安装java环境包

查询操作

Get /student/\_doc/\_search

Get /student/\_doc/\_search?q=Name:卯金刀

# Docker

命令：

docker --version 查看当前版本

docker ps 查看当前运行的容器

docker images 查看本地所有的镜像

docker run 运行一个容器

docker pull 用来拉取一个镜像到本地

1 docker logs 容器名称

2 docker attach 容器名称或容器ID

3 docker exec -it 59ec /bin/bash (如果提示“/bin/bash 未找到”之类的 可以改成:docker exec -it 59ec sh,退出ctrl+d)

4 docker stop 容器ID或容器名

5 docker kill 容器ID或容器名

6 docker rm 容器ID或容器名

7 systemctl restart docker  
8 docker image prune 删除所有镜像  
9.docker container prune 删除所有容器

运行docker容器，后台运行

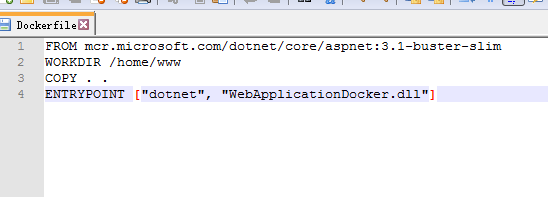
docker run -d -p 80:80 webapicore

docker stop 容器id

docker rm 容器id

部署CORE

1、上传Dockerfile文件



1. 项目上传后执行

docker build -t webapicore . 注意一个英文的“.”不能少

1. docker run -d -p 80:5006 webapicore

docker run -d -p 8001:8001 webapicore

查看端口监听情况

netstat -an | grep 8080

查找文件路径find / -name nginx.conf

编辑文件

vim nginx.conf

退出：先按【ESC】 再输入:q

常用操作指令http://www.51gjie.com/linux/1029.html